

REPORT OF TASK 2:-

AIM: - Create a model to predict the sign language by uploading sign language image and it should predict sign language easily. This model should sign language word as well. For an example "How are you" in sign language by uploading the video If we upload sign language video for "What is your Name" this model should predict this sign language as "Who are you" viceversa it should work also

Task to be complete for complete this project:-

- 1) Create a model that can predict the sign language on the bases of ASL (American Sign Language).
- 2) Create a GUI to use this model efficiently (on live video)

1) Creating a model that can predict the sign language on the bases of ASL (American Sign Language).

ASL Recognition Using Convolutional Neural Networks

1. Introduction

This project involves using a Convolutional Neural Network (CNN) to recognize American Sign Language (ASL) gestures. The model is trained on a dataset of images representing various ASL gestures and can predict the gesture shown in a new image.

2. Data Description

The dataset consists of images of ASL gestures, each belonging to one of 29 classes (A-Z, space, delete, nothing). The images are grayscale and of size 64x64 pixels.

3. Data Pre-processing

- **Image Data Generators:** ImageDataGenerators were used to rescale pixel values of images in both training and validation datasets by 1./255 to normalize the pixel values.
- **Batch Size:** A batch size of 64 was used to manage the computational load during training and validation.
- **Class Names:** The class names (A-Z, space, delete, nothing) were retrieved from the training data directory to ensure consistency in labeling.

4. Model Architecture

- **Convolutional Layers:** The model consists of four convolutional layers with ReLU activation functions. Each convolutional layer is followed by a max-pooling layer to reduce the spatial dimensions of the output and a dropout layer to prevent overfitting.
- **Fully Connected Layers:** After the convolutional layers, the model includes three fully connected (dense) layers. Each dense layer is followed by a dropout layer to further prevent overfitting.
- **Output Layer:** The final layer is a dense layer with 29 output units, corresponding to the 29 ASL classes, and uses a softmax activation function to provide probability distributions over the classes.

5. Model Training

- **Optimizer and Loss Function:** The model was compiled with an appropriate optimizer and loss function for multi-class classification.
- **TensorBoard:** TensorBoard was used to monitor the training process, including metrics such as loss and accuracy.
- **Epochs:** The model was trained for 100 epochs to ensure adequate learning of the features.

6. Model Evaluation

- **Training Accuracy:** The model achieved a training accuracy that indicates how well it learned the patterns in the training dataset.
- **Validation Accuracy:** The validation accuracy was monitored to assess the model's performance on unseen data, helping to detect overfitting.

7. Saving the Model

- **Model Structure and Weights:** The trained model's architecture and weights were saved for future use, enabling prediction on new data without retraining.
- **Files:** The model architecture was saved as a JSON file, and the weights were saved in an H5 file.

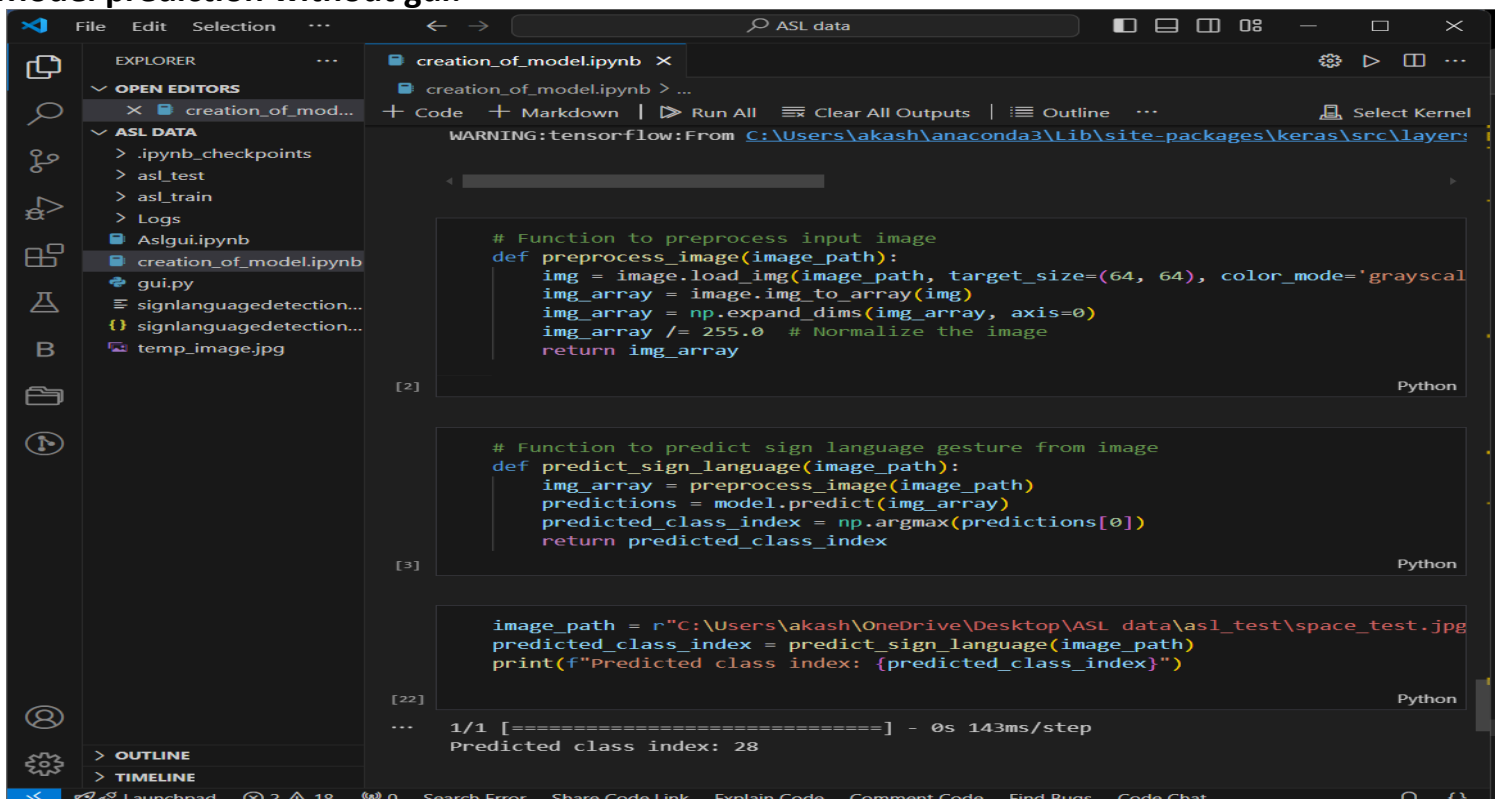
8. Prediction on New Data

- **Image Preprocessing:** A function was defined to preprocess new images by resizing and normalizing them, ensuring compatibility with the trained model.
- **Prediction Function:** A prediction function was implemented to load the preprocessed image into the model and output the predicted ASL gesture.

9. Conclusion

The CNN model demonstrated high accuracy in recognizing ASL gestures. The model and preprocessing functions were successfully implemented, allowing for accurate gesture predictions on new images. This system can be expanded for real-time ASL recognition applications.

#Model prediction without gui:-



```
WARNING:tensorflow:From C:\Users\akash\anaconda3\Lib\site-packages\keras\src\layer:

# Function to preprocess input image
def preprocess_image(image_path):
    img = image.load_img(image_path, target_size=(64, 64), color_mode='grayscale')
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0 # Normalize the image
    return img_array

# Function to predict sign language gesture from image
def predict_sign_language(image_path):
    img_array = preprocess_image(image_path)
    predictions = model.predict(img_array)
    predicted_class_index = np.argmax(predictions[0])
    return predicted_class_index

image_path = r"C:\Users\akash\OneDrive\Desktop\ASL_data\asl_test\space_test.jpg"
predicted_class_index = predict_sign_language(image_path)
print(f"Predicted class index: {predicted_class_index}")

1/1 [=====] - 0s 143ms/step
Predicted class index: 28
```

1) Create a GUI to use this model efficiently (on live video)

1. cv2 (OpenCV)

- **Purpose:** OpenCV is a computer vision library used for real-time computer vision tasks.
- **Usage:** Capturing video frames from a webcam (VideoCapture), resizing images, converting color spaces (cvtColor), and displaying images (imshow).

2. numpy

- **Purpose:** NumPy is used for numerical operations in Python.
- **Usage:** Handling arrays and manipulating image data efficiently, such as expanding dimensions (expand_dims), and normalizing pixel values.

3. keras

- **Purpose:** Keras is a high-level neural networks API, capable of running on top of TensorFlow.
- **Usage:** Loading and using a pre-trained deep learning model (load_model, predict).

4. tkinter

- **Purpose:** Tkinter is the standard GUI toolkit for Python.
- **Usage:** Creating a graphical user interface (Tk, Label, after for scheduling updates).

5. time

- **Purpose:** Provides various time-related functions.
- **Usage:** Introducing delays (sleep) in the video loop.

Functions

1. preprocess_frame(frame)

- **Purpose:** Preprocesses a video frame captured by the webcam.
- **Steps:**
 - Converts the frame to grayscale (cvtColor).
 - Resizes the frame to a fixed size (64x64 pixels).
 - Expands the dimensions of the frame to match the input shape expected by the model.
 - Normalizes pixel values to the range [0, 1].
- **Returns:** Preprocessed image frame as a numpy array.

2. predict(frame)

- **Purpose:** Predicts the ASL alphabet sign shown in the preprocessed frame using a loaded deep learning model.
- **Steps:**
 - Calls preprocess_frame to preprocess the input frame.
 - Uses the pre-trained model (model.predict) to obtain predictions.
 - Determines the predicted class index and maps it to a corresponding class label (class_labels).
- **Returns:** Predicted ASL alphabet sign as a string.

3. `update_sentence()`

- **Purpose:** Updates the GUI to display the current predicted sentence.
- **Steps:**
 - Configures the `sentence_label` to display the accumulated predicted sentence (`predicted_sentence`) joined into a string.
 - Uses `root.after` to schedule periodic updates (every 100 milliseconds).

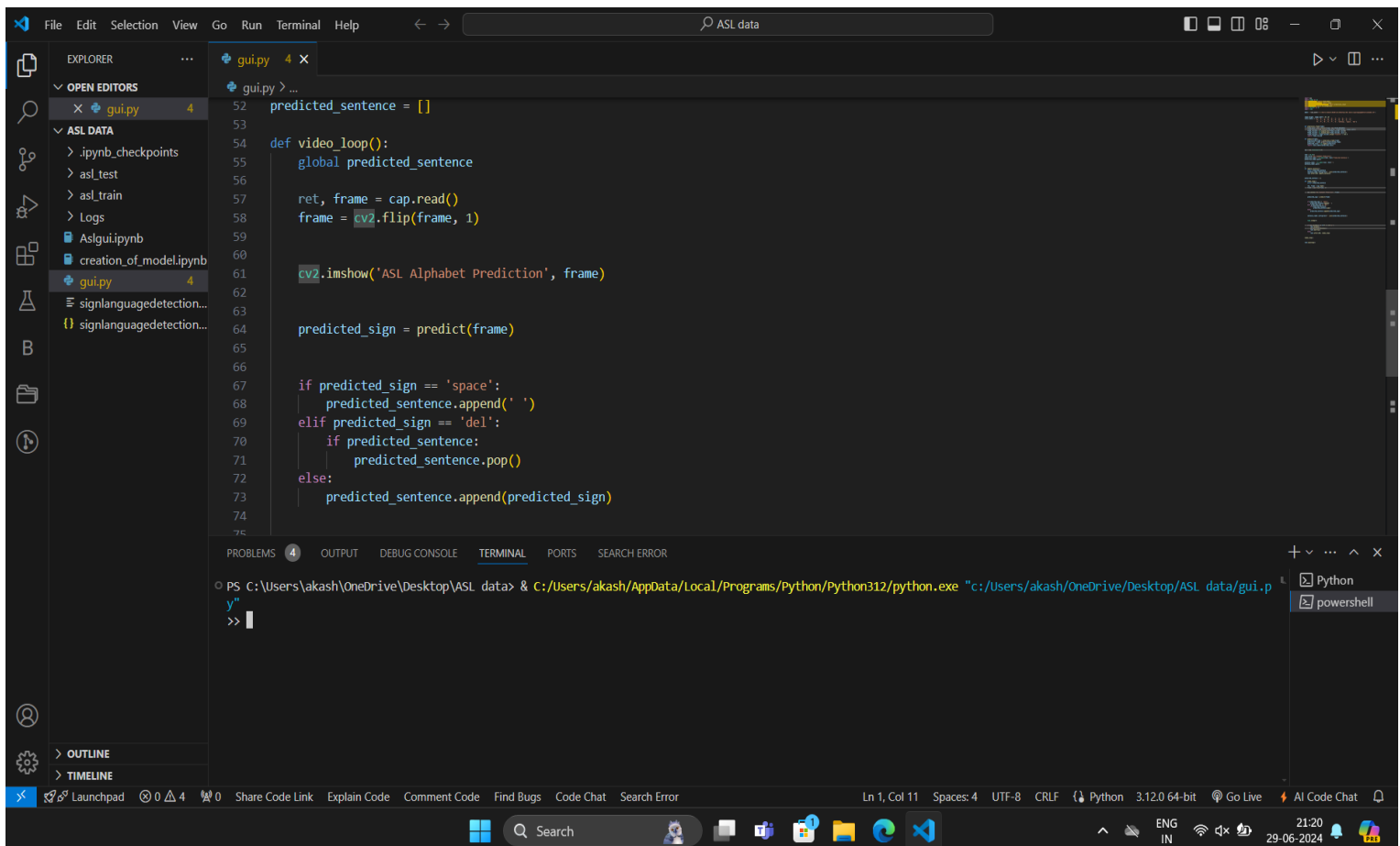
4. `video_loop()`

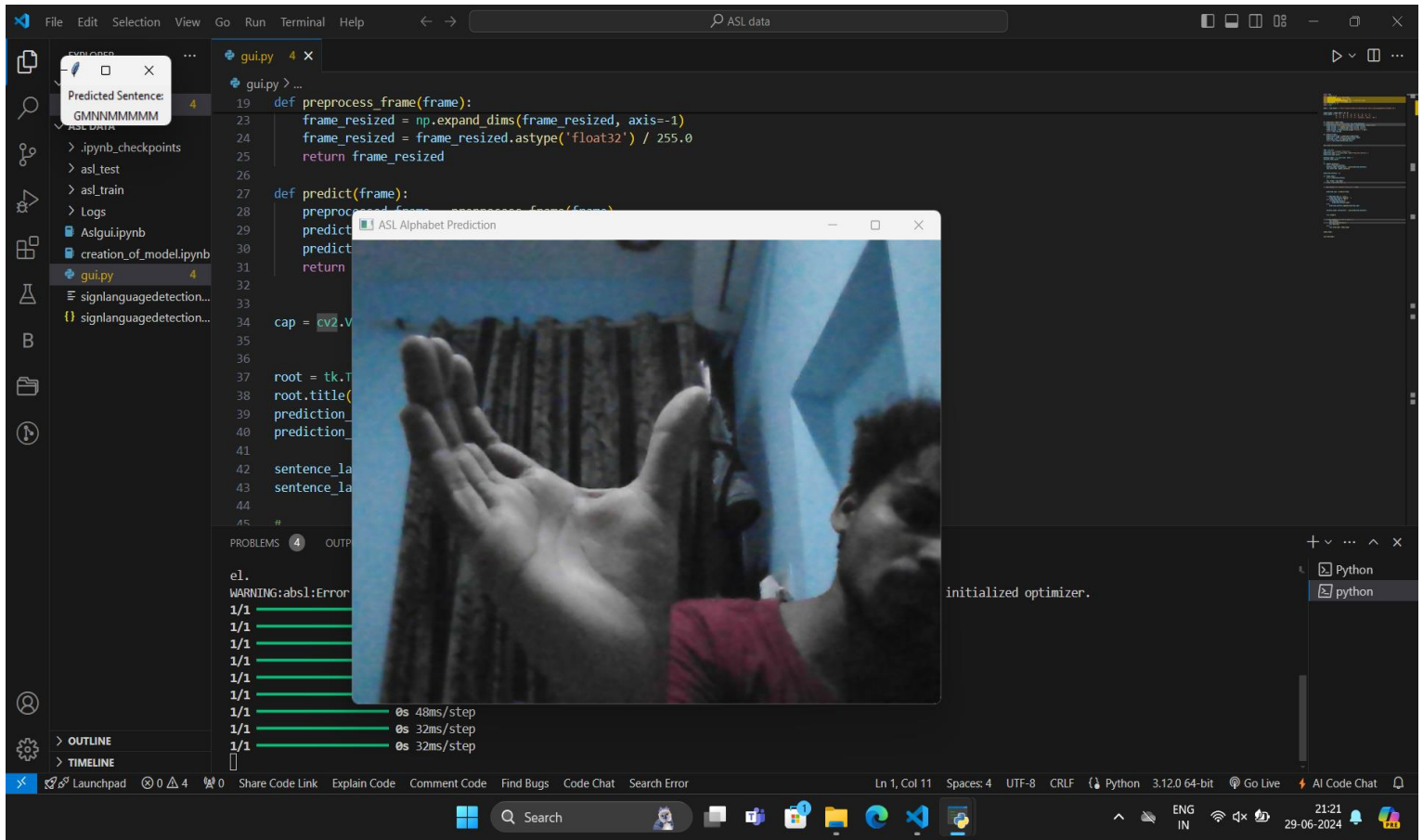
- **Purpose:** Continuously captures video frames from the webcam, performs predictions, and updates the GUI.
- **Steps:**
 - Uses `cap.read` to capture frames from the webcam.
 - Flips the frame horizontally (`cv2.flip`) for easier interaction.
 - Displays the video feed (`cv2.imshow`) with overlaid predicted signs.
 - Calls `predict` to predict the ASL alphabet sign in the frame.
 - Updates `predicted_sentence` based on the predicted sign.
 - Configures `sentence_label` to display the updated `predicted_sentence`.
 - Introduces a 3-second delay (`time.sleep`) between predictions.
 - Releases the webcam (`cap.release`), closes all OpenCV windows (`cv2.destroyAllWindows`), and terminates the GUI (`root.destroy`) when 'q' is pressed.

Conclusion

This script integrates real-time ASL alphabet prediction with a user-friendly GUI using OpenCV for video capture, Keras for deep learning model prediction, and Tkinter for graphical interface development. It enables interactive detection and display of ASL signs from live webcam video feeds, enhancing accessibility and user interaction.

#output:-





By ~ Akash kumar